

What is claimed is:

1. A computer system for transferring data between a receiving central processing unit (CPU) and a transmitting CPU by using only write operations, comprising:
- i) at least one receiving central processing unit (CPU);
 - ii) at least one transmitting CPU;
 - iii) a local memory for receiving CPU;
 - iv) a local memory for transmitting CPU;
 - v) means for connecting between receiving CPU and second CPU where such means transfer write operations faster than read operations; and
 - vi) a circular queue defined between designated addresses in said local memory of said receiving CPU.
2. A computer system for transferring data between a receiving central processing unit (CPU) and a transmitting CPU by using only write operations, according to claim 1, further comprising at least one receiving control register for control of said queue, allocated in said local memory of said receiving CPU.
3. A computer system for transferring data between a receiving central processing unit (CPU) and a transmitting CPU by using only write operations, according to claim 1, further comprising at least one register for control of said queue, said at least one register being allocated in said local memory of said transmitting CPU.
4. The system of claim 1 wherein said means for connecting between said CPUs is a PCI bus.

- 5 6. The system of claim 5 where said at least one control register in said receiving memory further comprises a total write register, which contains the total quantity of the data written into said queue of said receiving CPU by said transmitting CPU.
- 10 7. The system of claim 3, where said at least one control register in said transmitting memory further comprises a write head register, which contains a pointer to the location of the next write into said queue.
- 15 8. The system of claim 7 where said at least one control register in said receiving memory further comprises a transmitting total read register, which contains a second copy of the total quantity of data read from said queue by the receiving CPU.
- 20 9. The system of claim 5 where said at least one control register in said receiving memory further comprises a read head register, which contains a pointer to the location of the next read from said queue.
- 25 10. The system of claim 9 where said pointer for the next read from said queue and said pointer for the next write into said queue are set to point to the same address upon initialization.
11. The system of claim 10 where a maximum length is specified for said message to be written into said queue.

12. The system of claim 11 where a tail is added at the end of said queue equal to said maximum length for said message.

13. The system of claim 12 where a message separator is used to indicate the
5 end of said message.

14. The system of claim 13 where said message separator contains the length of said message.

10 15. The system of claim 13 where said message separator contains a "magic" number.

16. The system of claim 15 where, if said message separator contains an
erroneous "magic" number, an error message is generated.
15

17. The system of claim 13 where said message separator of the last message in said queue is a stopper designator, and is different from said message separator between messages.

20 18. The system of claim 17 where said stopper designator further contains a "magic" number.

19. The system of claim 18 where, if a stopper designator contains an erroneous "magic" number, an error message is generated.
25

20. A method for writing a data message into a receiving queue between memory segments separated by a data bus comprising the steps of:
i) checking that the length of said message is not greater than the length of said queue, and generating an error message if said message length is
30 greater than said queue length;

- ii) checking that length of said message is not greater than a maximum message length, and generating an error message if said message length is greater than said maximum message length;
- iii) repeatedly checking if there is sufficient memory available in said queue to contain said message until such time that sufficient memory is available;
- iv) writing said message into said queue;
- v) writing a stopper designator immediately after the end of said message;
- vi) replacing the stopper designator at the end of the immediately preceding message with a message separator; and
- vii) updating the content of a total writes register by adding the number of bytes written into said queue.

21. The method of claim 20 where said availability of memory in said queue is checked by a transmitting CPU by comparing said message length with the difference between said queue length and the difference between the total data written and total data read.

22. The method of claim 20 further comprising aligning said message to a predefined alignment scheme by adding alignment padding bytes at the end of said message.

23. The method of claim 22 where said alignment is on a four byte granularity.

24. The method of claim 22 further comprising assigning the new address of the write head by calculating the sum of the old write head plus said message length plus any applicable padding, plus the lengths of said beginning message separator and said stopper designator.

25. The method of claim 22 where, in the event that said data message was first written into the tail of said queue, calculating said new address of write head further comprises deducting said the queue length.

- 5 ~~26.~~ A method for reading a data message from a transmitting CPU into a queue with a tail of a receiving CPU, comprising:
- i) checking that said message to be read is in said queue or otherwise wait for said message to enter said queue;
 - 10 ii) checking that the "magic" number is valid, or otherwise generate an error message; and
 - 15 iii) reading said message without alignment padding bytes.

27. The method of claim 26, further comprising calculating the number of said alignment padding bytes added to said message.

28. The method of claim 27, further comprising calculating the new address of the read head as the sum of the old read head plus the length of the message, plus alignment padding, plus the lengths of the stoppers, in the event that said data read is not from said tail of said queue.

29. The method of claim 28 wherein said calculating further comprises deducting said length of said the queue, when said data was read from said tail of said queue,.

30. The method of claim 27 wherein said number of alignment padding bytes is added to the contents of the total read register.

31. The method of claim 30 wherein the content of the total read register is written into the corresponding register of the memory of said receiving CPU.

Handwritten signature and initials: "A2C AI" with a large checkmark.